Computers & Geosciences 93 (2016) 88–95

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Case study

pSIN: A scalable, *P*arallel algorithm for *S*eismic *IN*terferometry of large-N ambient-noise data

Po Chen *, Nicholas J. Taylor, Ken G. Dueker, Ian S. Keifer, Andra K. Wilson, Casey L. McGuffy, Christopher G. Novitsky, Alec J. Spears, W. Steven Holbrook

Department of Geology and Geophysics, University of Wyoming, United States

ARTICLE INFO

Article history:
Received 17 December 2015
Received in revised form
9 May 2016
Accepted 10 May 2016
Available online 11 May 2016

Keywords:
Seismic interferometry
Ambient-noise
Parallel algorithm
Message-passing interface

ABSTRACT

Seismic interferometry is a technique for extracting deterministic signals (i.e., ambient-noise Green's functions) from recordings of ambient-noise wavefields through cross-correlation and other related signal processing techniques. The extracted ambient-noise Green's functions can be used in ambient-noise tomography for constructing seismic structure models of the Earth's interior. The amount of calculations involved in the seismic interferometry procedure can be significant, especially for ambient-noise datasets collected by large seismic sensor arrays (i.e., "large-N" data). We present an efficient parallel algorithm, named *pSIN* (*P*arallel *S*eismic *IN*terferometry), for solving seismic interferometry problems on conventional distributed-memory computer clusters. The design of the algorithm is based on a two-dimensional partition of the ambient-noise data recorded by a seismic sensor array. We pay special attention to the balance of the computational load, inter-process communication overhead and memory usage across all MPI processes and we minimize the total number of I/O operations. We have tested the algorithm using a real ambient-noise dataset and obtained a significant amount of savings in processing time. Scaling tests have shown excellent strong scalability from 80 cores to over 2000 cores.

1. Introduction

It has been demonstrated, both experimentally and theoretically, that "by cross correlating and stacking the ambient noise recorded at two receivers, it is possible to recover the response of the material recorded at one receiver as if there were an impulse excitation at the other receiver" (i.e. the Green's function) (Rickett and Claerbout, 1999). Theoretical backgrounds of this principle have been developed using the normal-mode theory (e.g., Lobkis and Weaver, 2001), representation theorems (e.g., Weaver and Lobkis, 2004), time-reversal invariance (e.g., Derode et al., 2003), the principle of stationary phase (e.g., Snieder, 2004) and the reciprocity theorem (e.g., Wapenaar, 2004). This capability to extract deterministic response of the Earth from random noise is playing an increasingly important role in passive-source seismic tomography, since it allows us to exploit the density of the seismic network without waiting for natural earthquakes to occur.

Rapid advances in seismic data acquisition technology, in particular the availability of cable-free, autonomous geophones (e.g., Freed, 2008), have now opened up the possibility of recording the full ambient-noise wavefields and conducting ambient-noise tomography using large, dense 2D seismic arrays (e.g., Ritzwoller et al., 2011; Lin et al., 2013). Such full-wavefield analysis using dense seismic array data is sometimes called "large-N" seismic analysis. The number of autonomous receivers used in such large-N studies can be much larger than those used in conventional passive-source seismic experiments. For the ambient-noise tomography study in Long Beach, California documented in Lin et al. (2013), more than 5200 autonomous receivers recorded the ambient-noise wavefield for three weeks. For the Blair Wallis, Wyoming and Sierra Nevada, California critical-zone ambient-noise tomography study, we deployed 6 square arrays with about 400 autonomous receivers per array and each receiver recorded the ambient-noise wavefield for 3–4 days at a sampling rate of 500 samples per second, producing a dataset of about 1.5 TB.

To estimate ambient-noise Green's functions that can be used in ambient-noise tomography, the noise data recorded by a seismic array need to be processed following a sequence of operations that are often called "seismic interferometry". The computational cost of the entire procedure depends upon the total number of receivers N_r (Table 1) and the duration of the recording and can become a lengthy compute for large-N data. For the Blair-Wallis and Sierra-Nevada datasets used in this study, it took about 13 days of uninterrupted computing time on a single state-of-the-art four-core desktop computer to process the noise data recorded by one array of about 400 receivers. A natural choice for speeding up

* Corresponding author.
E-mail address: pchen@uwyo.edu (P. Chen).

http://dx.doi.org/10.1016/j.cageo.2016.05.003
0098-3004/

Table 1
List of symbols.

Symbol	Meaning
M	MPI process row number (Fig. 1)
N	MPI process column number (Fig. 1)
N_p	Total number of MPI processes, i.e., $N_p = M \times N$
$P_{i,j}$	The MPI process on the i-th row and j-th column (Fig. 1)
N_r	Total number of receivers of the seismic receiver array
N_r^p	Number of receivers per process row, i.e., $N_r^p = N_r / M$
N_{seg}	Total number of time segments (e.g., if the entire noise recording is 1-h long and each time segment is 2-min long, then N_{seg} = 1 h/2 min = 30)
N_{seg}^p	Number of time segments per process column, i.e., $N_{seg}^p = N_{seg}/N$
N_c	Total number of stacked cross-correlations, i.e., $N_c = (N_r - 1) \times N_r / 2$
N_s	Number of data samples per time segment
N_s^p	Number of data samples per MPI process, i.e., $N_s^p = N_r^p \times N_{seg}^p \times N_s$

the entire process is through parallelization.

The parallelization of the seismic interferometry algorithm appears to be trivial, as there is no apparent interdependence among the calculations of the stacked cross-correlations of different receiver pairs. A naive implementation is to execute the sequential seismic interferometry code on multiple CPU cores with each core calculating the stacked cross-correlations for a subset of all receiver pairs and no inter-process message passing is required. This problem seems to be "embarrassingly parallel". However, this naive implementation is heavily I/O-bound, especially on small to medium-sized computer clusters not equipped with powerful I/O subsystems. On most current-generation computer clusters, the memory size per core is limited. For our Blair-Wallis and Sierra-Nevada datasets, the average size of the binary file for one receiver is about 0.62 GB (each time sample is stored as a 4-byte single-precision float). On the Mount Moran cluster (a 284-node IBM System X cluster with each node having two 8-core Intel Xeon E5-2670 2.6 GHz processors) at the Advanced Research Computer Center (ARCC), University of Wyoming, the usable memory size per core is about 1.8 GB, which allows holding the recordings of a maximum of 3 receivers at one time, not counting the memory needed for storing other data during the calculation. To obtain all the stacked cross-correlations (approximately 80,000), each process needs to access the disk repeatedly during the entire calculation, which incurs heavy I/O overhead, especially at large core count.

One possibility for reducing the I/O overhead in the naive implementation is to adopt the Hadoop framework and the associated MapReduce computational paradigm and the Hadoop Distributed File System (HDFS). In fact, the ambient-noise seismic interferometry problem is an ideal candidate for the Hadoop implementation (Addair et al., 2014). When combined with the HDFS, the Hadoop framework leverages data locality by moving computation to the data, thereby providing extremely high I/O speed. However, on small to medium-sized computer clusters that are shared by many different types of applications, such as Mount Moran, the benefit for the ambient-noise seismic interferometry application may not justify the effort involved in setting up the Hadoop framework on the entire cluster.

In this study, we explore the possibility of reducing the I/O burden of ambient-noise seismic interferometry calculations on conventional distributed-memory computer clusters through algorithmic redesign. The *pSIN* code provided with this paper reads the ambient-noise data of the entire array from disk only once at the beginning of the execution and writes out all stacked cross-correlations at the end of the execution. Parallelization is implemented using the Message-Passing Interface (MPI). Computation, inter-process communication, and memory usage are well balanced across all CPU cores. Scaling tests using one of our Blair-Wallis and Sierra-Nevada datasets show excellent strong scalability from 80 cores to more than 2000 cores.

2. Algorithm

A widely adopted seismic interferometry technique (e.g., Bensen et al., 2007) involves three steps: single-receiver processing, inter-receiver cross-correlation and temporal stacking. At the single-receiver processing step, the entire time series recorded by each receiver is cut into N_{seg} equal-length time segments with N_s samples per segment (Table 1) and each time segment is normalized both in the time domain and in the frequency domain. At the inter-receiver cross-correlation step, each time segment at one receiver is cross-correlated with the corresponding time segment of every other receiver in the same array. For an array composed of N_r receivers, the total number of unique cross-correlation calculations is therefore $(N_r - 1) \times N_r \times N_{seg}/2$. At the temporal stacking step, all the N_{seg} cross-correlations for the same receiver pair are summed, producing $N_c = (N_r - 1) \times N_r/2$ stacked cross-correlations (Table 1), which are often called "ambient-noise Green's functions".

2.1. Single-receiver processing

Single-receiver processing is the first step in the seismic interferometry workflow. In this step, we need to read the ambient-noise data of the entire array from disk and decide the layout of the massive amount of noise data across all MPI processes. The data layout will determine the amount of inter-process communication overhead during the inter-receiver cross-correlation and temporal stacking steps, as well as the overhead for writing the stacked cross-correlations back to disk. The smallest data unit in the entire seismic interferometry process is one time segment of the noise data recorded by one receiver. We therefore need to consider how to group these smallest data units together so that the inter-process communication overhead in later steps can be minimized.

The smallest data unit has two natural coordinates: the index of the receiver that recorded this segment of noise data (i.e., receiver number) and the time-segment index that determines the offset from the beginning of the entire time series recorded by that receiver. We therefore adopt a two-dimensional Cartesian virtual topology to organize the MPI processes. The horizontal dimension is the time dimension with N MPI processes (Table 1) and the vertical dimension is the receiver dimension with M MPI processes (Table 1, Fig. 1). The total number of processes (i.e., cores) used in the entire calculation is therefore $N_p = M \times N$ (Table 1). The total number of receivers of a seismic array N_r is partitioned across the M rows of the process array and each row stores the noise data of $N_r^p = N_r/M$ receivers (Table 1). The entire time series of each receiver is partitioned across the N columns of the process array and each column stores $N_{seg}^p = N_{seg}/N$ time segments (Table 1). The total number of noise data samples per process is therefore $N_s^p = N_r^p \times N_{seg}^p \times N_s$ (Table 1), represented as black circles in Fig. 1.

On disk, the ambient-noise data are usually stored as one binary file per receiver, which contains the entire time series recorded by that receiver during the deployment, as well as some metadata (e.g., receiver ID and start time). The root rank of each process row, denoted as $P_{*,0}$ in Fig. 1, where "*" denotes any row rank, can read the binary files for different receivers simultaneously. After reading the binary file for one receiver, the root process of each row evenly distributes the data to the N processes in the same row using the MPI scatter function with each process receiving $N_{seg}^p \times N_s$ data samples. This reading/scattering procedure